Sobregeneración durante el Análisis Gramatical (*Overparsing*), Derivabilidad Parcial, Adyacencia y Propagación de Restricciones: el Algoritmo SCP

José F. Quesada
CICA (Centro de Informática Científica de Andalucía)
josefran@cica.es

Resumen

La sobregeneración durante el análisis gramatical (*overparsing*) es un problema que afecta a la mayoría de los algoritmos de parsing para gramáticas libres de contexto. Este trabajo introduce esta noción y estudia su influencia en los algoritmos de Earley, Kay (chart) y Tomita (GLR). El segundo objetivo consiste en analizar el comportamiento del algoritmo SCP ante este problema. Básicamente, el algoritmo SCP elimina la sobregeneración gramatical mediante la aplicación de la técnica de propagación de restricciones sobre un modelo formal basado en las relaciones de derivabilidad parcial y adyacencia.

1 Análisis Gramatical y Demostración Automática de Teoremas

La mayor parte de los sistemas de PLN necesitan un módulo de parsing (análisis gramatical). El objetivo fundamental de un parser consiste en la asignación de una o más estructuras a las sentencias, de acuerdo con un gramática dada. Desde un punto de vista operativo–funcional se puede concebir un parser como una función que recibe como entrada tanto la gramática como la cadena de palabras que debe analizar y que devuelve como salida la determinación de la gramaticalidad de la cadena de palabras (función de reconocimiento) y en caso de que la cadena pertenezca al lenguaje generado por la gramática se devuelve asimismo la estructura que dicha gramática asigna a la sentencia (función de análisis gramatical o parsing propiamente dicho).

Utilizando un enfoque más formal o logico–deductivo se puede comparar una gramática con una teoría, donde el símbolo raíz de la gramática es el único axioma y cada una de las reglas o producciones actuaría como una regla inferencial. Las cadenas de palabras se convertirían para este enfoque en potenciales teoremas, de forma que la tarea de un parser se puede describir como la determinación de la pertenencia de la cadena de palabras al espacio inducido o generado por la gramática a partir del axioma inicial. Siguiendo con esta comparación resulta evidente que las clásicas limitaciones formales en las reglas o producciones gramaticales que definen los niveles de las gramáticas regulares, libres de contexto, sensibles

al contexto y universales, se corresponden con limitaciones similares en la potencia generativa de las reglas de inferencia de las correspondientes teorías.

De esta forma el proceso de parsing se puede asimilar al de demostración automática de teoremas. Asimismo la similitud aparece al describir las dos estrategias básicas en cuanto a la dirección del parser o demostrador: ascendente o dirigido por los datos (cadena de palabras o teorema) y descendente o dirigido por la meta (símbolo raíz de la gramática o axiomas).

Estas similitudes nos servirán además para poner de manifiesto uno de los problemas más graves de los principales algoritmos de análisis gramatical o parsing: lo que hemos denominado *overparsing* o sobregeneración durante el proceso de análisis gramatical.

Reduciendo los procesos de demostración automática de teoremas y de análisis gramatical a una dimensión puramente simbólica o sintáctica (en el sentido de mera manipulación de símbolos) podemos observar que un lenguaje generado por una gramática (y de forma equivalente una teoría generada por un conjunto de axiomas y reglas de inferencia) no es más que un conjunto de cadenas de símbolos. Es decir, estaríamos ante un enfoque extensional de la noción de lenguaje (o teoría) según el cual una cadena de símbolos pertenece o no (en el sentido dicotómico de la teoría clásica de conjuntos) al lenguaje (o teoría).

Este nivel extensional o simbólico se corresponde con la función de reconocimiento indicada más arriba, y en él no tiene sentido la noción de ambigüedad: una cadena pertenece o no al lenguaje generado por la gramática.

No obstante, la idea de recursión o función recursiva en análisis formal de teorías y la idea de competencia lingüística inducen teorías y lenguajes infinitos recursivos (o incluso recursivamente enumerables) totalmente necesarios tanto desde un punto de vista teórico como práctico, pero que escapan a un tratamiento extensional. Es decir, las nociones de recursión (en el sentido que este término posee en teoría de la computabilidad) y la de competencia (en el sentido lingüístico fuerte de la filosofía del lenguaje) exigen la presencia de un modelo intensional de descripción de lenguajes y teorías, respectivamente gramáticas y conjuntos de axiomas–reglas de inferencia. A este nivel es digno de mención el paralelismo establecido entre las nociones de recursión y de competencia lingüística.

De esta forma, una gramática no es más que un modelo intensional de un lenguaje. Ahora bien, la imposición de estructura puede provocar ambigüedad. Recurriendo a la comparación logico–formal, una cosa es la gramaticalidad o demostrabilidad (veracidad) de un teorema y otra bien distinta es la estructura gramatical o demostración. La gramaticalidad o veracidad es ambivalente (si nos mantenemos en un nivel de lógica formal clásica que es el que aquí nos interesa). Por el contrario, la estructura gramatical y la demostración pueden ser múltiples para una misma sentencia o teorema. Sin embargo, aquí aparece una primera diferencia entre el ámbito lingüístico y el logico–deductivo. Mientras que en lógica estamos interesados en el valor de verdad (función de reconocimiento) y por tanto una sola demostración es suficiente para finalizar el proceso, en PLN suele interesar muy a menudo el nivel de estructura

gramatical (función de parsing) lo que exige un recorrido sistemático por todas las posibles demostraciones o estructuras gramaticales.

Hasta aquí nuestra discusión se ha mantenido en el mayor nivel posible de abstracción. No obstante, el hilo de la argumentación exige una delimitación más precisa, la cual se lleva a cabo en la siguiente sección.

2 Las Gramáticas Libres de Contexto en el Paradigma de la Unificación

La situación del PLN desde principios de la década de los 80 se puede caracterizar utilizando la nomenclatura de Thomas Kuhn como la aparación y consolidación del paradigma de la unificación (Shieber 1986). Bajo esta etiqueta se incluyen una serie de teorías y formalismos gramaticales entre cuyas características básicas se encuentran el ser fuertemente lexicalizados, asumir una orientación declarativa que separa el nivel de conocimiento del nivel inferencial u operativo, y la utilización del operador de unificación (definido a su vez sobre la relación de subsunción) como mecanismo básico composicional. Entre las principales propuestas que han aparecido merecen destacarse DCG (Pereira & Warren 1980), LFG (Bresnan 1982), FUG (Kay 1985), GPSG (Gazdar et al. 1985), PATR-II (Shieber 1986) y HPSG (Pollard & Sag 1994). Entre las propuestas que este paradigma destronó (al menos desde el punto de vista computacional) se encuentran las ATN (Woods 1970) y algunas versiones demasiado "puristas" (poco preocupadas por la eficiencia de la implementación) de la lingüística generativo–funcional (Grishman 1986).

En el contexto definido por lo que hemos denominado el paradigma de la unificación en PLN han aparecido dos enfoques básicos en lo que se refiere a la implementación del módulo de análisis gramatical (parsing):

- La primera opción ha consistido en la separación de una gramática de unificación en dos componentes. De un lado, aparece un núcleo libre de contexto y de otro lado aparece un nivel funcional implementado mediante ecuaciones de unificación asociadas con las reglas libres de contexto. Esta ha sido la alternativa elegida por DCG, FUG y LFG entre otros formalismos.
- La segunda opción, representada fundamentalmente por HPSG, consiste en reducir todo el proceso al nivel de unificación con lo que de alguna forma desaparece el módulo de parsing.

A favor de la segunda opción se pueden aducir razones de elegancia lingüística o uniformidad algorítmica. No obstante, teniendo en cuenta las complejidades algorítmicas de los módulos de parsing para gramáticas libres de contexto y de unificación para formalismos gramaticales de lenguajes naturales, la elección computacional guiada por el principio de eficiencia

real se decanta por la primera opción, siendo éste por tanto el enfoque que asumimos en este trabajo.

De esta forma, el problema del análisis gramatical planteado en la primera sección queda ahora limitado al ámbito de las gramáticas libres de contexto. Así pues, el siguiente aspecto que se debe abordar es el estudio de los principales algoritmos descritos en la literatura para este problema.

3 Algoritmos para el Análisis Gramatical de Lenguajes Libres de Contexto

A finales de los años 50 y principios de los 60 Noam Chomky realiza una serie de trabajos en los que se aborda la descripción formal del nivel sintáctico de los lenguajes (Chomsky 1956, 1959, 1962, 1963). De forma paralela, el comité para la especificación del lenguaje de programación ALGOL 60 (Dershem & Jipping 1990) introduce el formalismo BNF (Backus–Naur Form) a partir de los trabajos de John Backus (1957) en IBM y Peter Naur (1963). El formalismo BNF resultó ser equivalente al nivel 2 de las gramáticas libres de contexto propuesto por Chomsky.

Ambas corrientes de trabajo generaron puntos de interés diferentes. Desde el punto de vista del procesamiento del lenguaje natural el interés fundamental estaba en los lenguajes libres de contexto o modelos superiores (sensibles al contexto), mientras que desde el punto de vista de la teoría de compiladores el interés estaba en los lenguajes libres de contexto o modelos inferiores (regulares).

Un año después de la especificación BNF, Floyd (1964) introduce la noción de gramáticas con contexto limitado (*bounded context grammars*). Esta idea será utilizada inmediatamente por Knuth (1965) para la definición de las gramáticas $LR(k)$. Los problemas de eficiencia que presentan las técnicas para el tratamiento de las gramáticas LR son analizados por DeRemer en su tesis (DeRemer 1969), donde se propone como solución la utilización de dos subconjuntos de las gramáticas $LR(k)$: las gramáticas $SLR(k)$ y $LALR(k)$.

Así pues, esta línea de investigación parte de la noción original de gramática libre de contexto y evoluciona hacia modelos formalmente más restringidos para los cuales se pueden diseñar algoritmos eficientes de análisis. Estas técnicas serán usadas hasta nuestros días para el diseño del módulo sintáctico de los compiladores de lenguajes de programación.

En este mismo período comienzan a aparecer los primeros trabajos sobre el diseño de algoritmos generales de análisis de gramáticas libres de contexto. Así aparecen los algoritmos de Cocke (Hays 1962), Kasami (1965) y Younger (1967), que posteriormente han sido fusionados obteniéndose el conocido algoritmo CKY.

En la misma época presenta Earley su algoritmo (Earley 1968, 1970), el cual ha conseguido resistir de una forma más que favorable casi 30 años.

En los años 70 quizás lo más destacable es el desarrollo y consolidación de las técnicas LR, fundamentalmente LALR (Aho & Ullman 1972; Aho & Johnson 1974).

También en este período aparecen las técnicas conocidas como *Left Corner Parsing* a partir de un trabajo de Rosenkrantz y Lewis sobre *Deterministic Left Corner Parsing* (Rosenkrantz & Lewis 1970). Este modelo fue generalizado posteriormente por Kay (1989) bajo la etiqueta de análisis dirigido por núcleos (*Head Driven Parsing*). En un trabajo reciente sobre una implementación eficiente de un analizador *Head-Corner*, indica van Noord (1997):

> *Kay's presentation (1989) is reminiscent of the left–corner parser as presented by Pereira and Shieber (1987) which itself is a version without memoization of the BUP parser (Matsumoto et al. 1983).* (van Noord 1997), 3.

Las técnicas basadas en *left-corner driven* y *head-driven parsing* estan siendo muy usadas en los últimos años: (Satta & Stock 1989; Sikkel & den Akker 1993; Bouma & van Noord 1993). Van Noord (1994) aplica estas técnicas a las gramáticas de adjunción de árboles (TAGs).

Al principio de los años 80 se produce un revitalecimiento de la investigación en el campo del análisis sintáctico para gramáticas libres de contexto. Entre las motivaciones teóricas que justifican este nuevo interés se encuentra la "hipótesis del determinismo". Para muchos autores aparece una cierta paradoja entre el mecanismo aparentemente determinista que los hombres utilizan para la comprensión del lenguaje, y el no determinismo de todos los algoritmos de análisis. En términos formales, los hombres analizan sentencias en un tiempo linealmente dependiente de la longitud de la sentencia, mientras que la mayoría de los formalismos gramaticales, incluyendo los modelos libres de contexto, pueden asignar hasta un número de estructuras de análisis que crece exponencialmente con dicha longitud.

Guiado por esta intuición, en 1980 Mitch Marcus introduce lo que se ha denominado un formalismo no convencional que analiza de forma determinista las sentencias. La propuesta de Marcus se basa en una estrategia de tipo "esperar y mirar" (*wait and see*) construida sobre un parser de tipo LR.

A finales de los años 70, uno de los modelos dominantes en análisis sintáctico de lenguajes naturales son las redes de transición aumentadas (ATN). Fueron concebidas inicialmente por Woods (1970) como una extensión de las redes de transición recursivas mediante la incorporación de registros cuyo contenido puede ser controlado durante el recorrido de la red por instrucciones de asignación. Algunos autores reclamaron su utilidad como modelos psicolingüísticamente adecuados (Kaplan 1972) o incluso como herramientas útiles para implementar gramáticas de unificación como LFG mediante la utilización de los registros como variables lógicas. La principal crítica que ha recibido este formalismo es su inadecuación notacional, pues las redes, al aumentarse mediante registros dejan de ser modelos declarativos.

Partiendo de esta crítica, Fernando Pereira y David Warren proponen a principios de los 80 (Pereira & Warren 1980) el formalismo de las gramáticas de cláusulas definidas (DCG). Este formalismo se puede caracterizar por la expresión de las gramáticas como cláusulas de la lógica de primer orden, y más en concreto como cláusulas de Horn para un entorno Prolog. En la presentación de su trabajo, insisten Pereira y Warren en que este formalismo posee la misma potencia expresiva que las redes de transición aumentadas, a la vez que se añade mayor claridad, concisión y sobre todo, la utilización de Prolog como entorno de desarrollo.

En varios trabajos publicados a principios de los años 80, Martin Kay desarrolla lo que se conoce como técnicas de análisis sintáctico basadas en *charts*, o "tablas de subcadenas bien formadas" (Kay 1980, 1985).

Las aportaciones de Kay se pueden agrupar en dos bloques. En primer lugar, propone un algoritmo para el análisis sintáctico de gramáticas libres de contexto. De alguna forma, este algoritmo es una generalización de las técnicas surgidas previamente, entre las que destacan CKY, Earley y LR.

En segundo lugar, y quizás más importante, Kay va más allá de la presentación de un algoritmo y construye un modelo o esquema para el estudio de los algoritmos. Es importante recordar que el título de su trabajo es *"Algorithm Schemata and Data Structures in Syntactic Processing"*.

La idea propuesta por Kay acerca de "esquemas de algoritmos" sigue presente en este campo de investigación, y ha sido recogida por autores como Sikkel y Nijholt que en los últimos años han reelaborado aquella propuesta obteniendo un modelo formal de gran precisión y utilidad para el estudio de los algoritmos de análisis sintáctico (Sikkel & Nijholt 1997).

En la segunda mitad de los años 80, Masaru Tomita propone el algoritmo GLR (*Generalized LR*) (Tomita 1987, 1991).

Este algoritmo se puede describir como una extensión de las técnicas LR mediante el que se consigue un tratamiento eficiente de las gramáticas libres de contexto, el cual evita los problemas de la redundancia y del determinismo.

El algoritmo se basa en la utilización de una estructura de grafo (*graph–structured stack*) para la pila del autómata, lo que le permite llevar a cabo análisis alternativos (en paralelo) de una entrada y construir en la misma estructura los diferentes resultados parciales (*packet shared forest parse*).

En su versión original, el algoritmo de Tomita presenta serios problemas de completitud. De hecho, el algoritmo GLR no es completo ya que no puede tratar fenómenos tales como las gramáticas cíclicas o los fenómenos que aquí hemos denominado L–epsilon (*left–hyde recursion*). Esto ha generado una prolífica corriente de investigación en torno a la extensión del algoritmo GLR para lograr su completitud (Nozohoor–Farshi 1991; Nederhof & Sarbo 1993; Nederhof 1994; Nederhof 1996).

Sin embargo es en la década de los 90 cuando la investigación en el campo de las técni-

cas de análisis sintáctico de lenguajes naturales adquiere mayor énfasis. Como muestra de este interés basta mencionar la celebración bianual de un *International Workshop on Parsing Technologies*, iniciado en 1989.

En un intento de clasificación de los trabajos desarrollados durante los últimos años, se pueden destacar cuatro grandes líneas de trabajo:

- Técnicas probabilísticas. Aunque sus orígenes son anteriores a esta década, es cierto que ha sido en los últimos años cuando se ha realizado un trabajo mucho más sistemático y se han obtenido algunos resultados interesantes: (Corazza et al. 1991; Hindle & Rooth 1993; Magerman 1995; Briscoe & Carroll 1995; Collins 1996). No obstante, el gran problema de la utilización de técnicas probabilísticas en el análisis del lenguaje natural deriva de las propiedades formales de éstos métodos, que no son correctos ni completos.
- Paralelismo. Al igual que el resto de las áreas de la inteligencia artificial, en los últimos años han aparecido diferentes técnicas que permiten el análisis de lenguajes naturales usando arquitecturas paralelas como por ejemplo la propuesta de Nurkkala y Kumar (1994). (Alblas et al. 1994) contiene una bibliografía anotada sobre el tema. En este apartado merece destacarse el algoritmo de Rytter (1985) por su interés teórico.
- Reduccionismo. Un conjunto de trabajos sobre análisis sintáctico se pueden agrupar en torno a esta etiqueta ya que pretenden utilizar técnicas de menor potencia formal que las gramáticas libres de contexto, amparándose en el criterio de eficiencia. En este apartado se pueden citar la simulación de ciertos subconjuntos de las gramáticas libres de contexto mediante aproximaciones con estados finitos (Pereira & Wright 1996), la utilización de técnicas de poda y especialización gramatical (Rayner & Carter 1996) herederas de las propuestas acerca del análisis determinista, etc.
- Mejora de las técnicas disponibles y desarrollo de nuevos algoritmos. Es decir, multitud de trabajos están retomando los algoritmos ya disponibles, desde Earley hasta GLR pasando por *chart*, intentado modificaciones que mejoren sus prestaciones: (Nederhof & Sarbo 1993; Nederhof 1994; Erbach 1995; Bunt & Tomita 1996; Nederhof 1996). En este bloque merecen destacarse los múltiples trabajos que se han realizado con el objetivo de unir las técnicas de análisis sintáctico de gramáticas libres de contexto y las técnicas de unificación: (Shieber 1985; Haas 1989; Maxwell & Kaplan 1993; Wintner & Francez 1995), etc.

4 Overparsing: Sobregeneración durante el Análisis Gramatical

De entre los algoritmos descritos en la sección anterior creemos que los tres que mayor influencia han ejercido en el PLN son Earley, chart y GLR. Ahora bien, estos algoritmos presen-

tan un problema común que hemos denominado *overparsing*. Presentaremos este problema mediante un ejemplo y haciendo uso de la comparación con el ámbito logico–deductivo introducido en la primera sección.

Resumiendo lo dicho entonces, tenemos que en teoría de lenguajes formales, un lenguaje es un conjunto, y en teoría de conjuntos, un elemento pertenece o no a un conjunto. Es decir, un conjunto, y por tanto un lenguaje, es una estructura no ambigua. Una gramática puede ser considerada como una definición intensiva de un lenguaje. De esta forma, la noción de gramaticalidad se corresponde con la relación de pertenencia sobre un lenguaje (conjunto). Pero una gramática incorpora más información que el simple listado de los elementos del lenguaje (la especificación extensiva). Una gramática define una estructura: árbol o bosque de derivación, para cada elemento del lenguaje. La distancia entre la gramaticalidad y la estructura gramatical define un primer nivel de ambigüedad: *ambigüedad gramatical*, a la que denominaremos ambigüedad externa al analizador, puesto que está fuera de su alcance.

La siguiente noción que se debe tener en cuenta es el proceso mismo de análisis de una cadena de palabras de acuerdo con una gramática. Un parser debe ser capaz de determinar la relación de gramaticalidad (reconocimiento) y obtener la estructura gramatical (análisis), mediante un conjunto de operaciones, a las que nos referiremos como estructura del analizador.

La distancia entre la estructura gramatical y la estructura del analizador define un segundo nivel de ambigüedad: *ambigüedad del analizador*, también denominada ambigüedad temporal, y que aquí denominaremos ambigüedad interna al analizador, puesto que está provocada por él mismo. Denominaremos *overparsing* o sobregeneración gramatical a este trabajo propio del analizador pero innecesario desde el punto de vista del resultado final.

Consideremos la siguiente gramática:

```
S -> A1 b
S -> A2 c
A1 -> a
A1 -> a A1
A2 -> a
A2 -> a A2
```

y la siguiente cadena de palabras `a a a b`.

La figura 1 muestra los arcos generados por un analizador bidireccional basado en el algoritmo $chart$. Esta figura contiene los arcos creados en una primera fase donde sólo se habrán generado aquellos arcos que contienen al menos un símbolo terminal.

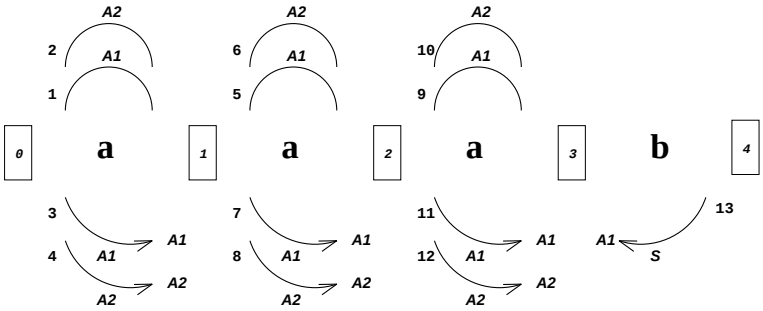

Figura 1: Ambigüedad del analizador (estado 1)

A continuación analizaremos lo que sucede en la posición 3. Existe una relación evidente entre los arcos `13` y `9`, pero los arcos `10`, `11` y `12` no poseen ningún enlace en esta posición. La figura 2 muestra el estado que se obtiene tras la eliminación de estos tres arcos.

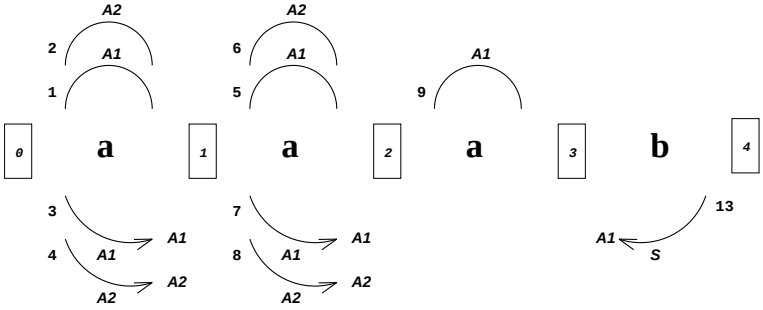

Figura 2: Ambigüedad del analizador (estado 2)

En la posición 2 existe una relación entre los arcos `7` y `9`, mientras que los arcos `5`, `6` y `8` pueden ser eliminados. Una vez que estos tres arcos han sido borrados, si analizamos la posición 1 podemos borrar los arcos `1`, `2` y `4` obteniendo el estado representado en la figura 3.

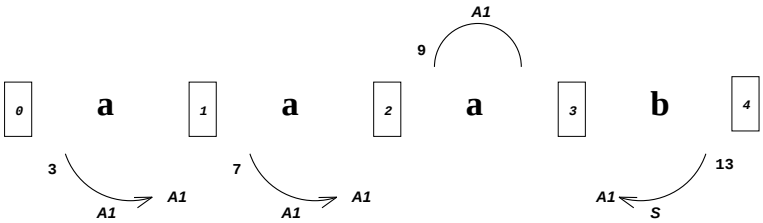

Figura 3: Ambigüedad del analizador (estado 3)

El problema ejemplificado para el algoritmo chart aparece asimismo en otros muchos algoritmos, y entre ellos en Earley y en GLR. Básicamente consiste en un exceso de trabajo por parte del parser. Es decir, éste crea más estructuras de las necesarias para construir el árbol o bosque de derivación gramatical de una sentencia (*overparsing*).

Teniendo en cuenta este problema así como otras motivaciones de naturaleza formal (corrección y completitud), de naturaleza computacional (eficiencia en la compilación y representación de gramáticas y en el proceso de análisis gramatical) y de naturaleza lingüística (máxima cobertura de fenómenos libres de contexto) se ha propuesto el algoritmo SCP de análisis sintáctico mediante propagación de restricciones (Quesada 1997a, 1997b).

El siguiente ejemplo muestra que el problema denominado *overparsing* no es trivial sino que condiciona decisivamente la eficiencia tanto algorítmica como computacional.

Consideremos la gramática G_{nm}:

```
(1:S ->N b)
(2:S ->M c)
(3:N->N a)
(4:N->a)
(5:M->M M)
(6:M->a)
```

Analizaremos cadenas de la forma: a^+b. Como se puede observar, dichas cadenas son no ambiguas según la gramática G_{nm}. Ahora bien, para determinar dicha no ambigüedad, cualquier algoritmo unidireccional (de izquierda a derecha[1]) debe construir un análisis (estructura `M`) que es exponencialmente ambiguo con relación al número de símbolos `a`. La complejidad computacional de los algoritmos CKY, Earley, chart y GLR es por tanto de orden n^3 donde n es la longitud de la cadena.

La siguiente figura representa gráficamente el trabajo que cualquiera de los algoritmos considerados debe realizar en la situación considerada, conforme van recibiendo los símbolos de la cadena de entrada:

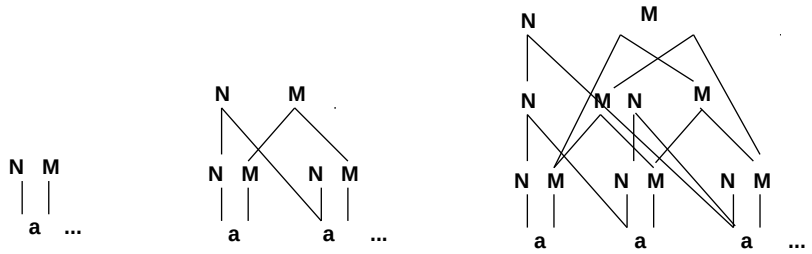

Figura 4: Parsing con la gramática G_{nm}.

Resumiendo, se puede demostrar que el trabajo útil de un parser para el problema anterior es de orden n mientras que el trabajo no útil (*overparsing*) es de orden $n^3 - n$.

[1] Si en lugar de un algoritmo unidireccional de izquierda a derecha se usa un algoritmo unidireccional en sentido contrario resulta elemental obtener una gramática donde se produce el mismo fenómeno que el descrito para G_{nm}.

5 El Algoritmo SCP

El algoritmo SCP (Quesada 1997a, 1997b) consigue evitar toda la sobregeneración gramatical descrita en la sección anterior. Para ello, la estrategia seguida se puede describir en dos niveles. En primer lugar el algoritmo trata de captar formalmente parte de la "inteligencia humana" aplicada para evitar la sobregeneración gramatical. En segundo lugar, es necesario aplicar eficientemente esta inteligencia formal. La primera parte se consigue mediante las relaciones de derivabilidad parcial y adyacencia, mientras que de la segunda se encargan las estrategias de propagación de restricciones, la representación basada en multi árboles virtuales y el análisis gramatical basado en estructuras *CaD* (colectores–difusores de información).

5.1 Inteligencia Formal para el Análisis Gramatical

A continuación presentamos la definición formal de las relaciones de derivabilidad parcial y adyacencia:

Sea $G =< N, T, P, S >$ una gramática libre de contexto, con $V = N \cup T$.

Sean $\alpha, \beta \in V$.

Definición 1 (Derivabilidad parcial por la izquierda) β es una derivación parcial por la izquierda de α:

$$\alpha \longmapsto_l^* \beta \quad syss \quad existen \quad \Gamma, \Delta, \Omega \in V^* \quad tales\ que \quad (\Gamma\alpha\Delta \Longrightarrow_G \Gamma\beta\Omega) \tag{1}$$

Definición 2 (Derivaciones parciales por la izquierda: $LPD(\alpha)$) Definimos el conjunto de derivaciones parciales por la izquierda de un símbolo α como sigue:

$$LPD(\alpha) = \{\beta \in V : \alpha \longmapsto_l^* \beta\} \tag{2}$$

Definición 3 (Derivabilidad parcial por la derecha) β es una derivación parcial por la derecha de α:

$$\alpha \longmapsto_r^* \beta \quad syss \quad existen \quad \Gamma, \Delta, \Omega \in V^* \quad tales\ que \quad (\Gamma\alpha\Delta \Longrightarrow_G \Omega\beta\Delta) \tag{3}$$

Definición 4 (Derivaciones parciales por la derecha: $RPD(\alpha)$) Definimos el conjunto de derivaciones parciales por la derecha de un símbolo α como sigue:

$$RPD(\alpha) = \{\beta \in V : \alpha \longmapsto_r^* \beta\} \tag{4}$$

Definición 5 (Adyacencia primaria) β es un adyacente primario de α:

$$\alpha \Uparrow \beta \quad syss \quad existen \quad \delta \in V \quad y \quad \Gamma, \Omega, \Delta \in V^* \quad tales\ que \\ \delta \longrightarrow \Gamma\alpha\Delta\beta\Omega \in P \quad y \quad E(\Delta) \tag{5}$$

Definición 6 (Adyacencia por la derecha) β es un adyacente por la derecha de α:

$$\alpha \Uparrow_r^* \beta \quad syss \quad existen \quad \gamma \in RPD(\alpha) \quad y \quad \delta \in LPD(\beta) \quad tales\ que \quad (\gamma \Uparrow \delta) \tag{6}$$

Definición 7 (Adyacencias por la derecha: $RA(\alpha)$) Definimos el conjunto de las adyacencias por la derecha de un símbolo α como sigue:

$$RA(\alpha) = \{\beta \in V : \alpha \Uparrow_r^* \beta\} \tag{7}$$

5.2 Propagación de Restricciones

En este apartado se describirá la idea de propagación de restricciones utilizando la gramática G_{nm}. En lugar de usar una estrategia unidireccional (como CKY, Earley, chart o GLR) el algoritmo SCP utiliza una estrategia bidireccional ascendente (Quesada & Amores Forthcoming). De esta forma, tras la recepción de una cadena de palabras, el algoritmo SCP crea las estructuras CaD, los nodos (a partir de la información devuelta por el nivel lexico–morfologico) y los eventos (disparados por las tablas de coberturas).

A continuación la fase de creación de enlaces (links) establece las relaciones entre eventos permitidas por las relaciones de derivabilidad parcial, adyacencia y fusión (Quesada 1997a). Estas relaciones garantizan formalmente la corrección y completitud del algoritmo de forma que se pueden eliminar los eventos que poseen algún extremo desconectado (no enlazado con otros eventos). Tras la eliminación de un evento, es posible que otros queden desconectados. De esta forma se consigue una propagación de restricciones no local mediante el análisis de relaciones locales, lo que supone un importante logro computacional.

Evidentemente las consecuencias a nivel de eficiencia suponen una gran mejora con respecto a otros algoritmos. La siguiente tabla muestra una comparación experimental entre SCP y el resto de algoritmos para el fenómeno que se está usando como ejemplo:

Longitud n	SCP n	CYK - Earley - chart - GLR n^3
1	0.0001	0.0001
4	0.0004	0.0064
16	0.0016	0.4096
64	0.0064	26.2144
256	0.0256	1677.7200 (27 minutos)
1024	0.1024	107374.0000 (29 horas)

6 Conclusión

Este trabajo aborda el problema de la sobregeneración durante el análisis gramatical (*overparsing*) en los algoritmos de parsing. Con este objetivo se discute en primer lugar el análisis gramatical desde una perspectiva logico–deductiva. A continuación se delimita el problema del análisis gramatical al paradigma del PLN basado en unificación, lo que nos permite circunscribir nuestro estudio al ámbito de los algoritmos para gramáticas libres de contexto, a cuyo análisis se dedica la sección 3. De esta especie de mapa de algoritmos hemos seleccionado tres estrategias básicas por su importancia en PLN: Earley, chart y GLR. No obstante, estos tres algoritmos muestran el fenómeno que hemos denominado *overparsing* a cuyo estudio se dedica la sección 4. Finalmente se estudia cómo el algoritmo SCP consigue resolver este problema. A grandes líneas, esto se logra mediante dos estrategias: en primer lugar, las relaciones de derivabilidad parcial y adyacencia captan la inteligencia formal que permite dirigir con todas las garantías formales el proceso de análisis; en segundo lugar, un mecanismo de propagación de restricciones actúa recursivamente aplicando las relaciones anteriores. La consecuencia es una mejora considerable tanto en la complejidad algorítmica como en la eficiencia real del proceso de análisis gramatical.

7 Bibliografía

Aho, A. V. & S. C. Johnson. 1974. LR Parsing. *Computing Surveys*, $6(2)$, Junio 1974, 99–124.

Aho, A. V. & J. D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling. Vol. I: Parsing.* Englewood Cliffs, N.J.: Prentice Hall.

Alblas, H., R. den Akker, P. O Luttighuis & K. Sikkel. 1994. A bibliography on parallel parsing. *SIGPLAN Notices*, $29(1)$, 54–65.

Backus, J. W. et al. 1957. The FORTRAN automatic coding system. *Proceedings of the Western Joint Computer Conference*, 188–198.

Bouma, G. & G. van Noord. 1993. Head–driven Parsing for Lexicalist Grammars: Experimental Results. En *Proceedings of the 6th Meeting of the European Chapter of the Association of Computational Linguistics,* Utrecht. 71–80.

Bresnan, J. ed. 1982. *The Mental Representation of Grammatical Relations.* Cambridge, MA: The MIT Press.

Briscoe, T. & J. Carroll. 1995. Developping and Evaluating a Probabilistic LR Parser of Part–of–Speech and Punctuation Labels. *Proceedings of the ACL/SIGPARSE 4th International Workshop on Parsing Technologies*, 48–58.

Bunt, H. & M. Tomita. eds. 1996. *Recent Advances in Parsing Technology*. Kluwer Academic Publishers.

Chomsky, N. 1956. Three models for the description of language. *PGIT*, **2(3)**, 113–124.

Chomsky, N. 1959. On certain formal properties of grammars. *Information and Control*, **2:2**, 137–167.

Chomsky, N. 1962. Context–free grammars and pushdown storage. *Quart. Prog. Dept. No. 65*, MIT Res. Lab. Elect., 187–194.

Chomsky, N. 1963. Formal properties of grammars. *Handbook of Math. Psych.*, **2**, New York: Wiley, 323-418.

Collins, M. J. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. Proceedings of ACL'96. También en *cmp-lg/9605012*.

Corazza, A., R. De Mori, R. Gretter & G. Satta. 1991. Computation of Probabilities for an Island–Driven Parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **13(9)**, 936–949.

DeRemer, F. L. 1969. *Practical Translators for LR(k) Languages*. Tesis doctoral. Harvard, Mass.: MIT.

Dershem, H. L. & M. J. Jipping. 1990. *Programming Languages: Structures and Models*.

Dowty, D. R., L. Karttunen & A. M. Zwicky. 1985. *Natural Language Parsing. Psycological, computational and theoretical perspectives.* Cambridge: Cambridge University Press.

Earley, J. 1968. *An Efficient Context–Free Parsing Algorithm.* Tesis doctoral, Carnegie–Mellon University, Pittsburg, PA.

Earley, J. 1970. An Efficient Context–Free Parsing Algorithm. *Communications of the ACM*, **13(2)**, 94-102.

Erbach, G. 1995. Bottom–Up Earley Deduction. *cmp–lg/9502004*.

Gazdar, G., E. Klein, G. K. Pullum & I. Sag. 1985. *Generalized Phrase Structure Grammar.* Cambridge: Harvard University Press.

Grishman, R. 1986. *Computational Linguistics*. Cambridge University Press.

Floyd, R. W. 1964. Bounded context syntactic analysis. *Commnunicatons of the ACM*, **7**, 62–67.

Haas, A. 1989. A Parsing Algorith for Unification Grammar. *Computational Linguistics*, **15(4)**, 219–232.

Hays, D. 1962. Automatic language–data processing. En H. Borko (ed.) 1962. *Computer Applications in the Behavioral Sciences*. Englewood Cliffs, NJ.: Prentice Hall.

Hindle, D. & M. Rooth. 1993. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, **19(1)**, 103–120.

Kaplan, R. M. 1972. Augmented transition networks as psychological models of sentence comprehension. *Artificial Intelligence*, **3**, 77–100.

Kasami, T. 1965. An Efficient Recognition and Syntax Analysis Algorithm for Context–Free Languages. Scientific Report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, Mass.

Kay, M. 1980. Algorithm Schemata and Data Structures in Syntactic Processing. *CSL-80-12 Xerox Palo Alto Research Center.*

Kay, M. 1985. Parsing in functional unification grammar. En (Dowty et al. 1985), 251–278.

Kay, M. 1989. Head Driven Parsing. *1st International Workshop on Parsing Technologies*, Pittsburgh, PA, 52–62.

Knuth, D. E. 1965. On the translation of languages from left to right. *Information and Control*, **8**, 607–639.

Magerman, D. 1995. Statistical Decision–Tree Models for Marsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 276–283.

Marcus, M. P. 1980. *A Theory of Syntactic Recognition for Natural Language.* Cambridge, Massachusetts: MIT Press.

Matsumoto, Y., H. Tanaka, H. Hirakawa, H. Miyoshi & H. Yasukawa. 1983. BUP: a bottom up parser embedded in Prolog. *New Generation Computing*, **1(2)**.

Maxwell III, J. T. & R. M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, **19(4)**, 571–590.

Naur, P. 1963. Revised Report on the Algorithmic Language Algol 60. *Comm. ACM*, **6:1**, 1–17.

Nederhof, M.-J. & J. J. Sarbo. 1993. Increasing the applicability of LR parsing. *Third International Workshop on Parsing Technologies*, 187–201.

Nederhof, M.-J. 1994. An optimal tabular parsing algorithm. *32nd Annual Meeting of the ACL*, 117–124.

Nederhof, M.-J. 1996. Efficient Tabular LR Parsing. *cmp-lg/9605018.*

Nozohoor-Farshi, R. 1991. GLR Parsing for ε-Grammers. En (Tomita 1991), 61–75.

Nurkkala, T. & V. Kumar. 1994. The Performance of a Highly Unstructured Parallel Algorithm on the KSR1. En *Proceedings of the Scalable High–Performance Computing Conference'94*, 215–220.

Pereira, F. C. N. & D. H. D. Warren. 1980. Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, **13**, 231–78.

Pereira, F. C. N. & S. M. Shieber. 1987. *Prolog and Natural Language Analysis*. Stanford: Center for the Study of Language and Information.

Pereira, F. C. N. & R. Wright. 1996. Finite–State Approsimation of Phrase–Structure Grammars. *cmp-lg/9603002*.

Pollard, C. J. & I. A. Sag. 1994. *Head–driven Phrase Structure Grammar*. Chicago: Chicago University Press.

Quesada, J. F. 1997a. *El algoritmo SCP de análisis sintáctico mediante propagación de restricciones. [The SCP parsing algorithm based on syntactic constraints propagation]*. Tesis Doctoral. July 1997. University of Seville.

Quesada, J. F. 1997b. A General, Sound and Efficient Natural Language Parsing Algorithm based on Syntactic Constraints Propagation. *Proceedings of the VII Conference AEPIA'97*, 775–786.

Quesada, J. F. & J. G. Amores. 1997. Linguistic and Computational Advantages of Bidirectional Bottom–Up Parsing with Top–Down Predictions. *Procesamiento del Lenguaje Natural*. **21**, 137–146.

Quesada, J. F. & J. G. Amores. (Forthcoming). *C for Natural Language Processing*. London: UCL Press.

Rayner, M. & D. Carter. 1996. Fast Parsing using Pruning and Grammar Specialization. Proceedings of ACL'96. También en *cmp-lg/9604017*.

Rosenkrantz, D. J. & P. M. Lewis. 1970. Deterministic Left Corner Parsing. *11th Annual Symposium on Switching and Automata Theory*, 139–152.

Rytter, W. 1985. On the recognition of context–free languages. *5th Symposium on Fundamentals of Computation Theory.* Lecture Notes in Computer Science, **208**, Springer–Verlag, 315–322.

Satta, G. & O. Stock. 1989. Head–driven bidirectional parsing. A Tabular method. *Proceedings of the Workshop on Parsing Technologies*. Pittsburg. 43–51.

Shieber S. M. 1985. Using Restriction to Extend Parsing Algorithms for Complex–Feature–Based Formalisms. En *Proceedings of the Association for Computational Linguistics*, 145–152.

Shieber S. M. 1986. *An Introduction to Unification–based Approaches to Grammar.* CSLI Lecture Notes 4. Stanford, California: Center for the Study of Language and Information.

Shieber S. M., F. C. N. Pereira, L. Karttunen & M. Kay. (eds.) 1986. *A Compilation of Papers on Unification–Based Grammar Formalisms. Parts I and II*. Report No. CSLI-86-48. Stanford, California: Center for the Study of Language and Information.

Sikkel, K. & R. op den Akker. 1993. Predictive head–corner chart parsing. *IWPT 3, Third International Workshop on Parsing Technologies*, Tilburg/Durbuy, 267–276.

Sikkel, K. & A. Nijholt. 1997. Parsing of Context–Free Languages. En G. Rozenberg & A. Salomaa. eds. 1997. *The Handbook of Formal Languages. Vol. II*. Berlin: Springer Verlag.

Tomita, M. 1987. An Efficient Augmented Context–Free Parsing Algorithm. $Computational\ Linguistics$ **13 (1-2)**, 31-46.

Tomita, M. 1991. *Generalized LR Parsing.* London: Kluwer Academic Publishers.

van Noord, G. 1994. Head Corner parsing for TAG. *Computational Intelligence*, **10:4**.

van Noord, G. 1997. An Efficient Implementation of the Head–Corner Parser. En *cmp-lg/9701004*.

Wintner, S. & N. Francez. 1995. Parsing with typed feature structures. *Proceedings of the 4th International Workshop on Parsing Technologies*, 273–287.

Woods, W. A. 1970. Transition network grammars for natural language analysis. *Communications of the ACM*, **13**, 591–606.

Younger, D. H. 1967. Recognition of context–free languages in time n^3. *Information and Control*, **10**, 189–208.